Experimental Security Analysis of SDN Network by Using Packet Sniffing and Spoofing Technique on POX and Ryu Controller

Yubaraj Gautam
Muroran Institute of Technology
Graduate School of Engineering
Muroran, Hokkaido, Japan
e-mail: 20043064@mmm.muroran-it.ac.jp

Bishnu Prasad Gautam
Kanazawa Gakuen University
Department of Economic Informatics
Kanazawa, Ishikawa, Japan
e-mail: gautam@kanazawa-gu.ac.jp

Kazuhiko Sato
Muroran Institute of Technology
Graduate School of Engineering
Muroran, Hokkaido, Japan
e-mail: kazu@mmm.muroran-it.ac.jp

***Abstract*—Software-Defined Networking (SDN) is an emerging network system which can configure and control the network by using programming technique through the specific controller (On the basis of Control Plane) to control whole network system. In this network system, the control plane and data plane are separated from each other through a specific controller such as Ryu, POX and OpenDayLight controller etc. In this network, the attacker could sniff or spoof the traffic by compromising SDN controllers and may utilize the entire network resources and may damage the entire network system which, in fact, should be disallowed by the controller. Therefore, in this research, we conducted an experiment to demonstrate how to mitigate such kinds of SDN attacks on both POX and Ryu controller separately to establish a secured network through a remotely operated SDN controller. In this research, we conducted two major experiments. Firstly, we conducted the layer 2 security on POX controller. Secondly, we conducted layer 3 security on Ryu controller. To analyze the layer 3 security functionalities of Ryu controller, we set some rules on the controller to filter the packets according to their packet type. Finally, we ensured that Ryu is one of the most comprehensive programmable controllers to provide the security features on SDN to develop firewall application in the future and offer future research direction.**

Keywords-Software-Defined Networking (SDN); Network Virtualization; Packet Sniffing; Packet Spoofing; SDN Security

I. Introduction

As the demand for information technology increases, the security issues also increase simultaneously [1]. Stable and easily manageable network is the first prioritized network system. In today's network, there are widely used physical network devices such as switch, and routers, firewall, etc. These devices have some limitations in functionality and difficult to add the new features in it. Management of a large network environment has become a significant issue in the traditional network [2]. Some common issues of conventional networks are frequent server down, network speed quality, data loss, etc. Because of these issues, it is very time consuming and difficult for a network administrator to troubleshoot the network in traditional settings. To overcome these issues, there is a new network system called Software-Defined Networking (SDN), which has a unique way to overcome those traditional network issues. It separates the network in the logical way of control plane and data plane. The control plane is also called the brain of the network. This means that it can control the whole network system. Data plane is the part of the network which carries user traffic. SDN handles almost every kind of network traffic through SDN applications. Therefore, it has many issues to solve, like SDN layer-wise security from the infrastructure layer to the application layer.

Network security is a very concerning topic in traditional and SDN network system also. In traditional network system, various security issues are yet to be solved. For example, DoS(Denial of Service) and DDoS(Distributed Denial of Service) attack, man in the middle attack, SQL injection, packet sniffing, and spoofing are the most common issues in the traditional network system, which are also the issues of the SDN network system. Furthermore, hackers can sniff our network system and illegally tap the network communication. Accessing network information without any permission could be very dangerous. The sniffed network packets could be manipulated and used for harmful purposes that is called network spoofing attack. SDN provides us different architecture to solve those issues [3]. Layer wise, we categorized into three parts. The first layer contains the security of the infrastructure layer; the second is the security of the control plane, and the third is the security of the application layer. In this research, we have conducted an experiment on python-based POX and Ryu controller to identify the best controller for the purpose of SDN network security. We designed a network architecture to experiment on both controller and analyze by sending the different types of packets from one host to another through the controllers. The objective of this research is to analyze the both controller in order to develop a firewall application by using packet sniffing and spoofing technique in SDN. In our proposed SDN network architecture, the network packets will allow to enter into the network or drop according to the policy files or rules by the SDN controller.

II. Literature Review

SDN system is an emerging network system for the general user and the network administrator also. It has centralized controlling, faster and programmable features, which were lacking in traditional network system. However, it has high chances to be attacked by the attacker in various ways also. For example, the attacker could easily spoof new flows in the controller and would forward specific types of traffic that should be rejected across the network. To minimize such kind of attack, we have developed a prototype application that can handle Sniffing and spoofing attack in SDN network. Gautam and Shrestha [4] have presented a model for cloud computing security which are proposed in a layer based architecture. The authors propose different solutions and security policies to promote a common level of understanding between the users, business communities, and necessary security requirements for the Jyaguchi application. SDN can provide a unique way to solve this problem, however there are some issues in SDN security itself. Thus, we are developing an SDN application to minimize those security issues. The vulnerability of network traffic in data centers under various kinds of attacks research [5] provide us some detail of SYN flood and DNS attack monitoring and analysis of network traffic by using TCP dump and Wireshark. The vulnerability of network traffic in the data center under various kinds of attacks like DoS/DDoS attack is the biggest issue in traditional network system. To minimize this, there were some experiments and analysis to reduce such kinds of attacks by highlighting the major security threats based upon SYN floods followed by currently faced real working scenario by giving an example for DNS attack. It is also a fact that many organizations have not adequately secured there DNS servers. To minimize such kings of problems, the administrators are required to update their knowledge to face the dynamicity of attackers and prepare with a countermeasure. Network monitoring is a challenging task for every network administrator. If the network system is centralized, then the analysis of network traffic will be easier to do. Providing a security policy for such a network system is our research objective. Previously, we have done some experiments on SDN system by developing the SDN hub application into different manners and checked the quality of service (QoS) based on its bandwidth, latency, and packet loss [6]. The limitation of this research was that we had not considered the security issues in the SDN control plane. In the research of ‘Validating of User Flows to Protect SDN Environments’ [7], the authors propose an SDN design with star topology. They have argued that by using SDN multi-controller, it is effective to secure the network environment which is the better architecture for preserving or holding time than other works in the literature. In the SDN network system, DoS and DDoS attacks are the major issues like traditional network. The article of ‘simulating dos attack in SDN attack in SDN network using POX controller and Mininet emulator’ [8] provides the solution of the defending system for attack where the number of packets could either be high or low. The article of the effect of input-output buffering to minimize flow control blocking [9] analyze the different input-output buffering strategies which affect flow control blocking in a SDN system. The network architecture of the SDN based 5G network system [10] with a centralized security controller that communicates with the SDN controller has been conducted previously. There are some relevant researches in the SDN system security [11][12]. The authors are mentioning different applications for working at different layers to block unwanted packets. However, we realized that there is still a gap in SDN security if we analyze it through layer-wise security perspective. In this research, we would like to focus on layer two and layer three security in the SDN (i.e. control plane) briefly to provide more protection by using ARP sniffing and spoofing detection algorithms.

III. Research Issues

In previous research [13], the authors have compared between Ryu and NOX controller to compare the behavior and performance of these controllers. The performances of applications in the python based controller POX, Ryu and Pyretic have tested by Karamjeet Kaur, Sukhveer Kaur and Vipin Gupta [14] on the basis of round trip time (RTT) and throughput. However, they have not tested these controllers from the perspective of security performance. Thus, we have found gap in the literature specifically in preventing SDN layer 2 security for packet sniffing and spoofing attacks. In this attack, an unauthorized user can get the network information through packet sniffing. This attack also called the passive security attack because it gathers only the network information. If the attacker uses the gathered information and manipulates that information to gain access to the network system for an illegal purpose, that is called a spoofing attack. In such an attack, there is no satisfactory solution in SDN and conventional network systems. In this research, we are proposing how to mitigate such kinds of attacks in the SDN network system by using programmable SDN controller.

IV. Proposed Solution

We propose a layered based security architecture that can overcome the security issue of SDN network. We argue that by providing packet evaluation scheme on the basis of security rules and policy in each layer, security vulnerabilities of SDN network can be reduced sharply. Particularly, we would like to develop an application implemented in python programming language for the SDN control plane. To protect SDN network from a packet sniffing attack, we implemented an application into the SDN control plane and conducted a number of experiments by sniffing and spoofing the packets. We defined some rules and policies into the program by which the incoming packets are compared as per the firewall rules. If the packet header matches with the defined rules the packet is allowed to flow into the network. Otherwise, our firewall program rejects it. To match the rules, we use the layer two frame header entities like source mac, destination mac, source port, destination port, source IP, destination IP, etc.

In some cases, the sniffed data is used to exploit the targeted host by using DoS attack. To address such issues, it is needed that a system that can alert the administrator if controller detect suspicious packets. To achieve this goal, we compare the usual flow of the packets with the current flow and if the current flow is exceeding the normal rate of flow, the application may decide this situation as the abnormalities pattern in the network. If the condition does not match with the defined rules, the system will send the alert to the administrator otherwise, it will allow to access network.

V. Research Methodology

This network architecture can filter network packets by comparing their packet header to the defined rules into the controller. To achieve our goal, we would like to follow the Figure 1 steps.

We developed an algorithm to implement the above flow chart into the application program. The first method is to create the network topology. Then, we send the network packets from one host to another through the controller. The controller will check whether it is layer two packets or not. If incoming packets are for layer two, it will allow for further processing, whether it is acceptable packets or not; otherwise, it will drop the packets. For further processing, our running application has predefined rules and compare those rules to the incoming packets header. If the packet header matches with application rules, then the packet will pass into the network by the controller. Otherwise, it will drop the packets.

```
Preparations:
Step 1: Start controller
Step 2: Create network topology by accessing controller remotely
Step 3: Send packets from one host to another through controller
Step 4: Controller check the incoming packets header.
Algorithm:
Step 5: Validation process start
        If (arp request)
                {
                If (matches the rules)
                        {
                        Connection Successful
                        }
                Else
                        {
                        Invalid packet
                        }
        Else
                {
                Invalid packet
                }
Step 6: end
```

In this research, we are implementing the method of packet sniffing detection system into the SDN control plane. As we can see in Figure 1, if the controller gets the packets from the network host, it will check the packet's header. If the packet header is under the defined rules of the controller, it will pass. Otherwise, it will drop the packets.

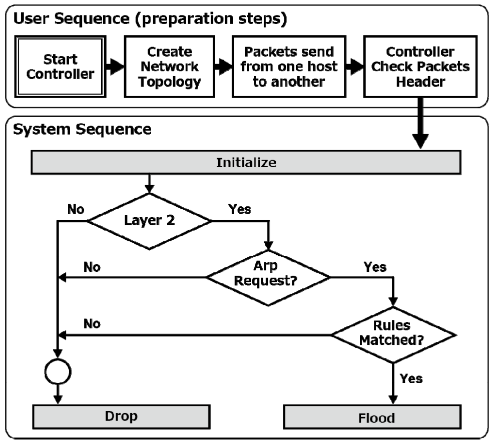

Figure 1: Sequence Diagram for Packet Sniffing Defending System

VI. Experimental Setup

In our lab experimental setup, we have used the following tools to do experiment and analyze packet by using sniffing and spoofing algorithm.

VirtualBox: VirtualBox is an open source hypervisor which helps us to create a virtual environment in a single computer. To create our network topology, we are using VirtualBox in windows and installing ubuntu (with Mininet) as a guest OS.

Mininet: This is the SDN network emulator to create network topology. We are using two nodes of Mininet; Mininet CLI, which is used for running POX controller and Mininet GUI for creating SDN network topology. We access the controller machine remotely and analyze the performance of the network.

Wireshark: It is a tool for capturing the network packets. We are using this tool to analyze the network traffic and its performances.

For the experiments, four hosts are connected to the SDN switch, and the switch is connected to the controller. Packet sniffing and spoofing application will be running into the POX controller, as shown in Figure 2.

In this scenario, host 1(10.0.0.1), host 2 (10.0.0.2), host 3 (10.0.0.3) and host 4 (10.0.0.4) have assigned simple mac address from 00:00:00:00:00:01 to 00:00:00:00:00:04 respectively. All the hosts are connected to the SDN switch and switch is connected to the controller. The SDN application layer (firewall application) and control layer (POX controller) is executed in PC1 and infrastructure layer (switch and hosts) are created in the PC2 and both are connected remotely as we can see in the Figure 2.

VII. Experimental Analysis

We divided the experiments into the two parts. The first experiment is done by using the application without providing MAC address and the second experiment is done with detecting MAC addresses.

A. *Experiment without filtering rules*

In this experiment, the controller allows the packets to enter the network because the rules are not set. In this setting, it could cause sniffing and spoofing attack to the network if the controller has no any packet filtering rules are set. We observed the flow pattern in this setting and compare it after applying the filtering rules.

B. *Experiment with filtering rules*

For this experiment, we allow the controller to filter the network packets on the basis of packets header. We used source MAC and destination MAC to identify whether the incoming packets are trustable to the network or not.

For this purpose, the packet detection application is running into the controller. This application selects the network packets according to the policy file as we can see in the Table 1.

In order to conduct our experiment, the controller is accessing remotely with network topology having four hosts and a single virtual switch. We created such kind of SDN network topology in the GUI machine with the following command.

```
mn --controller=remote, ip=10.0.0.100, port=6633 --mac
--topo single, 4
```

This command creates the topology as shown in Figure 2 having one switch, four hosts, and one controller where the controller is working remotely. After creating network topology, the flow of packets has added to the switch according to the policy file.

This command created us one switch, four hosts, and one controller where the controller is working remotely.

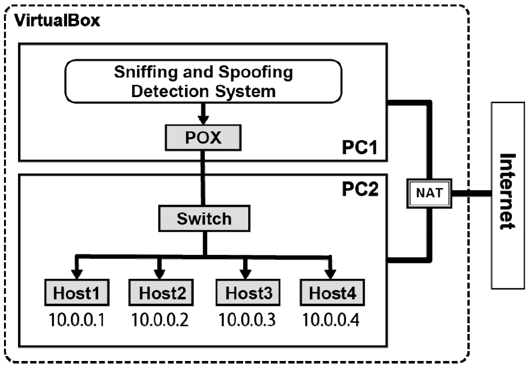

Figure 2: Network Simulation Scenario

Table 1: Policy Files of MAC Address

S. N	Source MAC	Destination MAC	Action
1	00:00:00:00:00:01	00:00:00:00:00:03	Deny
2	00:00:00:00:00:02	00:00:00:00:00:04	Deny

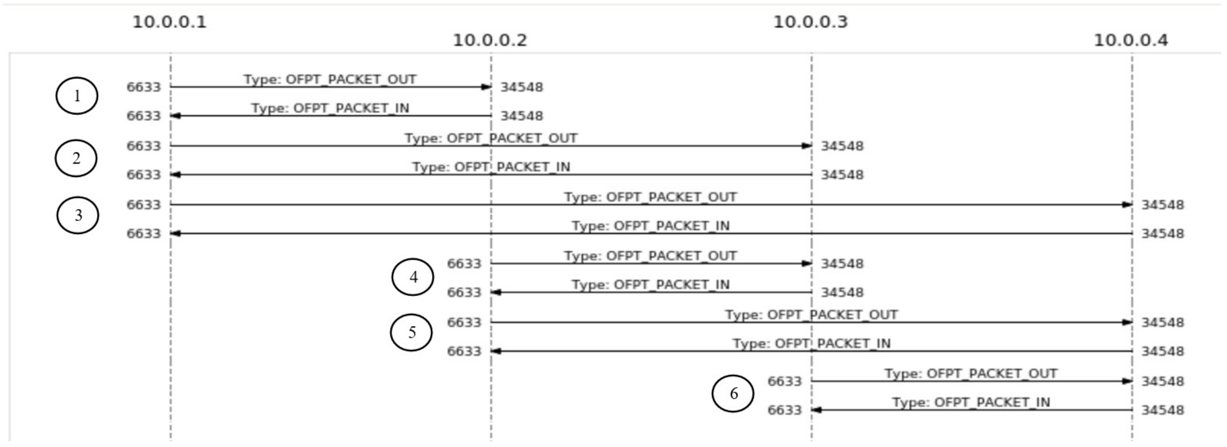

Figure 3: Packets Flow without Filtering Rules

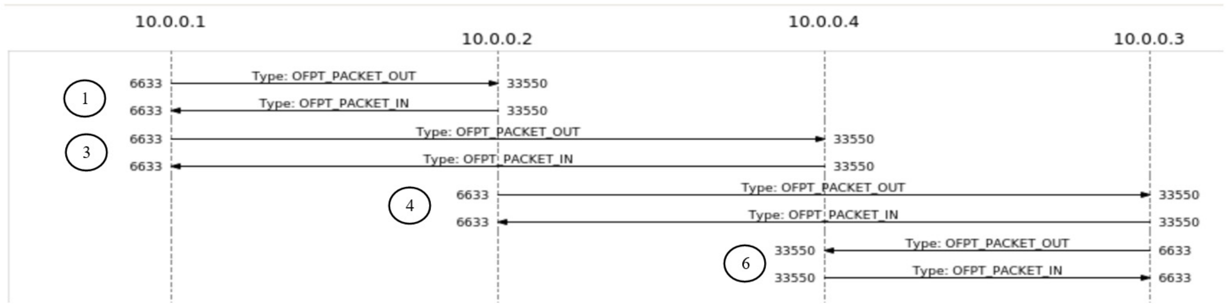

Figure 4: Packets Flow with Filtering Rules

After creating network topology, the flow of packets has added to the switch according to the policy file.

As shown in the Table 1, if the source MAC address is 00:00:00:00:00:01 and destination MAC address is 00:00:00:00:00:03, then the controller dropped the packets. Similarly, if source MAC address is 00:00:00:00:00:02, and destination MAC address is 00:00:00:00:00:04, then the controller will drop the packets. Otherwise, it will flood the packet.

C. *Analysis of experiments using wireshark*

We analyzed the packet flow by using Wireshark tool. Wireshark captures the flow of packets into the network system. Figure 3 is a flow observed without filtering rules. It shows that all the hosts are permitted to connect to each other using packet-in and packet-out method. This includes 6 interactions, No.1 to No.6. On the other hand, Figure 4 shows the packet flow taken after the settings of packet filtering in the POX controller. In this setting, No.1, 3, 4 and 6 are connected successfully and No. 2 and No.5 are rejected from the controller by using filtering rules.

In our experiments and analysis, we can clearly see in Table 2 that the network architecture having our application is safer because the application detects the packet's header and select the packets through mac addresses if those network devices are not allowed by the controller. For example, if such nodes are flooding unwanted packets in a network, then network administrator can prevent from network congestion discarding the incoming packets from such nodes as done in our experiment.

We have done some experiments on POX controller to provide the layer two security for SDN network. It has some difficulties to provide layer-wise security on POX controller. Therefore, we have done some other experiment on Ryu controller by using inbuilt *rest_firewall* application. We designed a network architecture on Ryu controller as shown in the Figure 5. Ryu is used as SDN controller. In the experimented topology, there are one victim host (H2) and two attacker hosts (H1, H3) which are in the same network

Table 2: Experimental Analysis using POX Controller

Hosts	Experiment 1	Experiment 2
H1 to H2	✓	✓
H1 to H3	✓	X
H1 to H4	✓	✓
H2 to H1	✓	✓
H2 to H3	✓	✓
H2 to H4	✓	X
H3 to H1	✓	X
H3 to H2	✓	✓
H3 to H4	✓	✓
H4 to H1	✓	✓
H4 to H2	✓	X
H4 to H3	✓	✓

Table 3: Rules for Packets Sharing

S. N	Allow Packets
1	curl -X POST -d '{"nw_src": "10.0.0.1/32", "nw_dst": "10.0.0.2/32", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001
2	curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.1/32", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001
3	curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.3/32", "nw_proto": "TCP"}' http://localhost:8080/firewall/rules/0000000000000001
4	curl -X POST -d '{"nw_src": "10.0.0.3/32", "nw_dst": "10.0.0.2/32", "nw_proto": "TCP"}' http://localhost:8080/firewall/rules/0000000000000001
	Deny Packets
7	curl -X POST -d '{"nw_src": "10.0.0.1/32", "nw_dst": "10.0.0.2/32", "nw_proto": "TCP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001
8	curl -X POST -d '{"nw_src": "10.0.0.1/32", "nw_dst": "10.0.0.2/32", "nw_proto": "UDP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001
9	curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.1/32", "nw_proto": "TCP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001
10	curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.1/32", "nw_proto": "UDP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001
11	curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.3/32", "nw_proto": "ICMP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001
12	curl -X POST -d '{"nw_src": "10.0.0.2/32", "nw_dst": "10.0.0.3/32", "nw_proto": "UDP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001
13	curl -X POST -d '{"nw_src": "10.0.0.3/32", "nw_dst": "10.0.0.2/32", "nw_proto": "ICMP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001
14	curl -X POST -d '{"nw_src": "10.0.0.3/32", "nw_dst": "10.0.0.2/32", "nw_proto": "UDP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/0000000000000001

connected with the switch. All the network architecture is designed into the virtual environment.

For this experiment, we set rules to the controller as shown in Table 3. The controller will analyze the incoming packet's source IP, destination IP and packet type to identify the trustable packets to the network. If the attacker floods the packets to the victim, the controller will compare with packet's header entities to the defined rules and drop all the prohibited packets to minimize the flooded packets quantity.

According to our rules, host 1 and host 2 can share only the ICMP packets and host 2 and host 3 can share only the TCP packets. Rest of the communications will deny from the controller.

We did four experiments on our network system and analyze by flooding the different types of packets from attackers to the victim host. We conducted an experiment by flooding the ICMP and TCP packets up to 4000 packets (1000 packets for each experiment) from attacker hosts to the victim host. As a result, the controller detects the flooded packets type and drop the packets according to the rules in the controller. It drops the 50% of flooded packets by allowing only 2000 packets into the network as we can see in Table 4.

VIII. Conclusion and Future Works

In this research, we conducted two major experiments and analyzed to identify the most comprehensive SDN controller in order to develop the firewall application for the security purposes. On POX controller, we analyzed the performances by using the layer two firewall application to block the unwanted packets according to the policy file. On the other hand, Ryu controller already has verities of applications sample to use as a reference application. To analyze the performance of Ryu controller, we used *'rest_firewall.py'* application and set the rules to filter the packets according to their packets type. In our conclusion, Ryu controller provides better performance to provide security features for SDN network than the POX controller.

In our future work, we aim to develop a firewall application on Ryu controller which will detect the malicious packets and drop the packets if those packets are harmful to the network. Specially, the application will protect from the sniffing and spoofing attack and provide the layer-wise security for the SDN network system. Furthermore, we would like to develop an alert system to announce administrator if unusual packets detect from the SDN controller. In order to protect SDN network comprehensively, it is needed that the layer two security should also be strengthened thereby considering the frame header entities like source mac, destination mac, source port, destination port, VLAN ID, source IP, destination IP, etc. However, our experiment at layer 2 does not support the VLAN packets and thus remained this task for our future work. In our future work, we will do comparative study between Ryu and OpenDayLight controller as well.

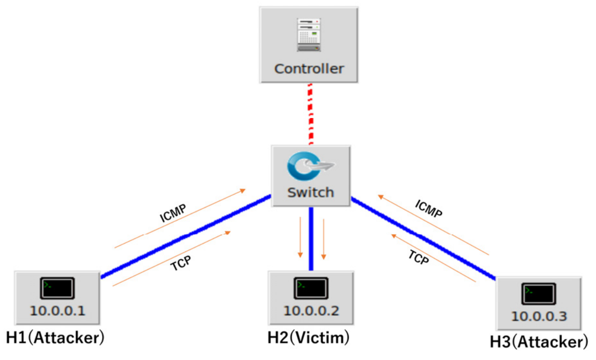

Figure 5: Lab Scenario for Ryu Controller

Table 4: Experimental Analysis

Experiments	attacks	Packet Type	Total Packets	Allowing packets
1	H1 to H2	TCP	1000	X
2		ICMP	1000	✓
3	H3 to H2	TCP	1000	✓
4		ICMP	1000	X

References

[1] M. V. O. De Assis, M. P. Novaes, C. B. Zerbini, L. F. Carvalho, T. Abrao, and M. L. Proenca, "Fast Defense System Against Attacks in Software Defined Networks," *IEEE Access*, October, 2018.

[2] M. Ikram Lali *et al.*, "The Nucleus Performance Evaluation of Software Defined Networking vs. Traditional Networks," *Nucl.*, no. January, 2017.

[3] S. Scott-Hayward, "Trailing the Snail: SDN Controller Security Evolution," November, 2017.

[4] B. P. Gautam and D. Shrestha, "A model for the development of Universal Browser for proper utilization of computer resources available in service cloud over secured environment," *Proc. Int. MultiConference Eng. Comput. Sci.*, no. March, 2010.

[5] D. Pun, A. Batajoo, and B. P. Gautam, "Vulnerability of Network Traffic in Data Centers under Various kinds of Attacks," 2015.

[6] Y. Gautam, B. P. Gautam, and H. Asami, "Quality of Service (QoS) SDN Application Program Interface (API) Using POX Controller," 2019.

[7] I. H. Abdulqadder, D. Zou, I. T. Aziz, and B. Yuan, "Validating User Flows to Protect Software Defined Network Environments," *Secur. Commun. Networks*, no. February, 2018.

[8] W. H. Muragaa, K. Seman, and M. F. Marhusin, "Simulating DDoS Attack in sdn Network Using POX Controller and Mininet Emulator," 2018.

[9] M. I. Lali, M. M. Bilal, M. S. Nawaz, M. Deen, B. Shahzad, and S. Khaliq, "Effect of Input-Output (IO) Buffering to Minimize Flow Control Blocking in Software Defined Networking," 2016.

[10] X. Liang and X. Qiu, "A software defined security architecture for SDN-based 5G network," *Proc. 2016 5th Int. Conf.*, 2017.

[11] D. He, S. Chan, and M. Guizani, "SECURING SOFTWARE DEFINED NETWORK AGAINST ROGUE CONTROLLERS," *IEEE Commun. Mag.*, no. December, 2016.

[12] A. M. Abdelsalam, A. B. El-sisi, and V. R. K, "Mitigating ARP Spoofing Attacks in Software-Defined Networks," in *ICCTA 2015, At Alexandria, Egypt*, 2015.

[13] S. Asadollahi, B. Goswami, and M. Sameer, "Ryu Controller's Scalability Experiment on Software Defined Networks."

[14] K. Kaur, S. Kaur, and V. Gupta, "PERFORMANCE ANALYSIS OF PYTHON BASED OPENFLOW CONTROLLERS," 2016.